PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ○ The ACM Digital Library [✓ The Guide]

software testing by decomposition <and> broken into subset

THE GUIDE TO COMPUTING LITERATURE

Feedback Report a problem Satisfaction survey

Terms used
software testing by decomposition and **broken into subset**

Found **184,823** of **825,846**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The Digital Library

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Software development processes: A proposed approach to process decomposition and collaboration for MARVEL
Andrew Z. Tong
October 1993 **Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: software engineering - Volume 1**
Full text available: pdf(920.68 KB) Additional Information: full citation, abstract, references

An industrial-scale software process consists of a collection of subprocesses run by different people either simultaneously or at different times. This paper introduces a process decomposition and collaboration model to determine what the user and designer of such a process need from a Process-Centered Environment (PCE), and proposes extensions to a centralized rule-based PCE, MARVEL 3.1, to address some of those needs. This approach has a number of potential benefits such as support for various ...

2 A schema for interprocedural modification side-effect analysis with pointer aliasing
Barbara G. Ryder, William A. Landi, Philip A. Stocks, Sean Zhang, Rita Altucher
March 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 2
Full text available: pdf(1.72 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The first interprocedural modification side-effects analysis for C (MODC) that obtains better than worst-case precision on programs with general-purpose pointer usage is presented with empirical results. The analysis consists of an algorithm schema corresponding to a family of MODC algorithms with two independent phases: one for determining pointer-induced aliases and a subsequent one for propagating interprocedural ...

3 Operating System Structures to Support Security and Reliable Software
Theodore A. Linden
December 1976 **ACM Computing Surveys (CSUR)**, Volume 8 Issue 4
Full text available: pdf(3.49 MB) Additional Information: full citation, references, citings, index terms

4 On randomization in sequential and distributed algorithms
Rajiv Gupta, Scott A. Smolka, Shaji Bhaskar
March 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 1
Full text available: pdf(8.01 MB) Additional Information: full citation, abstract, references, citings, index terms

Probabilistic, or randomized, algorithms are fast becoming as commonplace as conventional deterministic algorithms. This survey presents five techniques that have been widely used in the design of randomized algorithms. These techniques are illustrated using 12 randomized algorithms—both sequential and distributed— that span a wide range of applications, including:primality testing (a classical problem in number theory), interactive probabilistic proof s ...

Keywords: Byzantine agreement, CSP, analysis of algorithms, computational complexity, dining philosophers problem, distributed algorithms, graph isomorphism, hashing, interactive probabilistic proof systems, leader election, message routing, nearest-neighbors problem, perfect hashing, primality testing, probabilistic techniques, randomized or probabilistic algorithms, randomized quicksort, sequential algorithms, transitive tournaments, universal hashing

5 Compiler transformations for high-performance computing
David F. Bacon, Susan L. Graham, Oliver J. Sharp
December 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 4

Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citings, index terms, review

In the last three decades a large number of compiler transformations for optimizing programs have been implemented. Most optimizations for uniprocessors reduce the number of instructions executed by the program using transformations based on the analysis of scalar quantities and data-flow techniques. In contrast, optimizations for high-performance superscalar, vector, and parallel processors maximize parallelism and memory locality with transformations that rely on tracking the properties o ...

Keywords: compilation, dependence analysis, locality, multiprocessors, optimization, parallelism, superscalar processors, vectorization

6 Computing curricula 2001
September 2001 **Journal on Educational Resources in Computing (JERIC)**

Full text available: pdf(613.63 KB) html(2.78 KB) Additional Information: full citation, references, citings, index terms

7 Combining Software and Hardware Verification Techniques
Robert P. Kurshan, Vladimir Levin, Marius Minea, Doron Peled, Hüsnü Yenigün
November 2002 **Formal Methods in System Design**, Volume 21 Issue 3

Full text available: Publisher Site Additional Information: full citation, abstract, references, index terms

Combining verification methods developed separately for software and hardware is motivated by the industry's need for a technology that would make formal verification of realistic software/hardware co-designs practical. We focus on techniques that have proved successful in each of the two domains: BDD-based symbolic model checking for hardware verification and partial order reduction for the verification of concurrent software programs. In this paper, we first suggest a modification of partia ...

Keywords: formal verification, hardware/software co-design, model checking, partial order reduction

8 Active zones in CSG for accelerating boundary evaluation, redundancy elimination, interference detection, and shading algorithms
Jaroslaw R. Rossignac, Herbert B. Voelcker
November 1988 **ACM Transactions on Graphics (TOG)**, Volume 8 Issue 1

Full text available: Additional Information: full citation, abstract, references, citings, index pdf(2.67 MB) terms, review

Solids defined by Boolean combinations of solid primitives may be represented in constructive solid geometry (CSG) as binary trees. Most CSG-based algorithms (e.g., for boundary evaluation, graphic shading, interference detection) do various forms of set-membership classification by traversing the tree associated with the solid. These algorithms usually generate intermediate results that do not contribute to the final result, and hence may be regarded as redundant and a source of inefficien ...

9 Manageable object-oriented development: abstraction, decomposition, and modeling
John A. Anderson, John D. Sheffler, Elaine S. Ward
December 1991 **Proceedings of the conference on TRI-Ada '91: today's accomplishments; tomorrow's expectations**
Full text available: pdf(1.70 MB) Additional Information: full citation, references, citings, index terms

10 Distributed nested decomposition of staircase linear programs
James K. Ho, R. P. Sundarraj
June 1997 **ACM Transactions on Mathematical Software (TOMS)**, Volume 23 Issue 2
Full text available: pdf(199.07 KB) Additional Information: full citation, abstract, references, index terms, review

This article considers the application of a primal nested-decomposition method to solve staircase linear programs (SLPs) on distributed-memory, multiple-instruction-multiple-data computers. Due to the coupling that exists among the stages of an SLP, a standard parallel-decompositon algorithm for these problems would allow only a subset of the subproblem processes to overlap with one another at any give time. We propose algorithms that seek to increase the amount of overlap among the process ...

Keywords: computational linear programming, distributed computation

11 A concurrency analysis tool suite for Ada programs: rationale, design, and preliminary experience
Michal Young, Richard N. Taylor, David L. Levine, Kari A. Nies, Debra Brodbeck
January 1995 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 4 Issue 1
Full text available: pdf(2.93 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Cats (Concurrency Analysis Tool Suite) is designed to satisfy several criteria: it must analyze implementation-level Ada source code and check user-specified conditions associated with program source code; it must be modularized in a fashion that supports flexible composition with other tool components, including integration with a variety of testing and analysis techniques; and its performance and capacity must be sufficient for analysis of real application programs. Meeting these objectiv ...

Keywords: Ada, concurrency, software development environments, static analysis, tool integration

12 Concrete Type Inference: Delivering Object-Oriented Applications
Ole Agesen
January 1996 Technical Report, Sun Microsystems, Inc.
Full text available: pdf(684.36 KB) Additional Information: full citation, abstract

A dissertation submitted to the Department of Computer Science and the Committee on Graduate Studies of Stanford University in partial fulfillment of the requirements for the degree of Doctor of Philosophy. (December 1995) *Note: This document is ~1.6Mb*

13 An Integrated Approach to Designing and Evaluating Collaborative Applications and Infrastructures
Prasun Dewan
January 2001 **Computer Supported Cooperative Work**, Volume 10 Issue 1
Full text available: Publisher Site Additional Information: full citation, abstract, index terms

Collaborative systems include both general infrastructures and specific applications for supporting collaboration. Because of the relative newness and complexity of these systems, it has been unclear what approach should be used to design and evaluate them. Based on the lessons learned from our work and that of others on collaborative systems, we have derived an integrated approach to researching collaborative applications and infrastructures. The approach can be de ...

14 Dynamic Network Flow with Uncertain Arc Capacities: Decomposition Algorithm and Computational Results
Gregory D. Glockner, George L. Nemhauser, Craig A. Tovey
March 2001 **Computational Optimization and Applications**, Volume 18 Issue 3
Full text available: Publisher Site Additional Information: full citation, abstract, index terms

In a multiperiod dynamic network flow problem, we model uncertain arc capacities using scenario aggregation. This model is so large that it may be difficult to obtain optimal integer or even continuous solutions. We develop a Lagrangian decomposition method based on the structure recently introduced in G.D. Glockner and G.L. Nemhauser, Operations Research, vol. 48, pp. 233–242, 2000. Our algorithm produces a near-optimal primal integral solution and an optimum solution to the Lagrangian ...

Keywords: decomposition, dynamic, network flow, stochastic

15 Inside a software design team: knowledge acquisition, sharing, and integration
Diane B. Walz, Joyce J. Elam, Bill Curtis
October 1993 **Communications of the ACM**, Volume 36 Issue 10
Full text available: pdf(4.84 MB) Additional Information: full citation, references, citings, index terms, review

Keywords: case study, empirical studies of software development, requirements determination, software design teams, software management

16 Apel: A Graphical Yet Executable Formalism for Process Modeling
S. Dami, J. Estublier, M. Amiour
January 1998 **Automated Software Engineering**, Volume 5 Issue 1
Full text available: Publisher Site Additional Information: full citation, abstract

Software process improvement requires high level formalisms for describing project-specific, organizational and quality aspects. These formalisms must be convenient not only for capture but also for execution purposes. In order to fulfill these requirements and to build a software process environment capable of supporting engineering tasks we have designed a new graphical, but still enactable, formalism called APEL (for Abstract Process Engine Language).

APEL is very ambitious in ...

17 How Software Engineering Tools Organize Programmer Behavior During the Task of Data Encapsulation
Robert W. Bowdidge, William G. Griswold
March 1997 **Empirical Software Engineering**, Volume 2 Issue 3

Full text available: Publisher Site Additional Information: full citation, abstract

Tool-assisted meaning-preserving program restructuring has been proposed to aid the evolution of large software systems. These systems are difficult to modify because relevant information is often widely distributed. We performed an exploratory study to determine how programmers used a restructuring tool interface called the "star diagram" to organize their behavior for the task of encapsulating a data structure. We videotaped six pairs of programmers while the ...

Keywords: data encapsulation, empirical study, restructuring, software tools

18 A structural view of the Cedar programming environment
Daniel C. Swinehart, Polle T. Zellweger, Richard J. Beach, Robert B. Hagmann
August 1986 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 8 Issue 4

Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents an overview of the Cedar programming environment, focusing on its overall structure—that is, the major components of Cedar and the way they are organized. Cedar supports the development of programs written in a single programming language, also called Cedar. Its primary purpose is to increase the productivity of programmers whose activities include experimental programming and the development of prototype software systems for a high-performance personal computer. T ...

19 Program Understanding as Constraint Satisfaction: Representation and Reasoning Techniques
Steven Woods, Qiang Yang
April 1998 **Automated Software Engineering**, Volume 5 Issue 2

Full text available: Publisher Site Additional Information: full citation, abstract

The process of understanding a source code in a high-level programming language involves complex computation. Given a piece of legacy code and a library of program plan templates, understanding the code corresponds to building mappings from parts of the source code to particular program plans. These mappings could be used to assist an expert in reverse engineering legacy code, to facilitate software reuse, or to assist in the translation of the source into another programming langua ...

20 Reverse engineering and system renovation—an annotated bibliography
M. G. J. van den Brand, P. Klint, C. Verhoef
January 1997 **ACM SIGSOFT Software Engineering Notes**, Volume 22 Issue 1

Full text available: pdf(1.32 MB) Additional Information: full citation, abstract, index terms

To facilitate research in the field of reverse engineering and system renovation we have compiled an annotated bibliography. We put the contributions not only in alphabetical order but also grouped by topic so that readers focusing on a certain topic can read their annotations in the alphabetical listing. We also compiled an annotated list of pointers to information about reverse engineering and system renovation that can be reached via Internet. For the sake of ease we also incorporated a brief ...

Keywords: annotated bibliography, reverse engineering, system renovation

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ○ The ACM Digital Library ● The Guide

software testing by decomposition <and> broken into subset

THE GUIDE TO COMPUTING LITERATURE

Feedback Report a problem Satisfaction survey

Terms used
software debug by decomposition and **broken into submodule**

Found **168,350** of **825,846**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The Digital Library

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 A visual software process language
Terry Shepard, Steve Sibbald, Colin Wortley
April 1992 **Communications of the ACM**, Volume 35 Issue 4
Full text available: pdf(1.03 MB) Additional Information: full citation, references, citings, index terms, review

Keywords: configuration management, life cycle management, process programming, software process

2 Testing Software Requirements with Z and Statecharts Applied to an Embedded Control System†Ö†1
Hye Yeon Kim, Frederick T. Sheldon
September 2004 **Software Quality Control**, Volume 12 Issue 3
Full text available: Publisher Site Additional Information: full citation, abstract

Software development starts by specifying the requirements. A Software Requirements Specification (SRS) describes what the software must do. Naturally, the SRS takes the core role as the descriptive documentation at every phase of the development cycle. To avoid problems in the latter development phases and reduce life-cycle costs, it is crucial to ensure that the specification is correct. This paper describes how to model, test and evaluate (i.e., check, examine, and probe) a natural languag ...

Keywords: Statecharts, Z, completeness, consistency, fault-tolerance, requirements specification and validation

3 Recomposition: Coordinating a Web of Software Dependencies
Rebecca E. Grinter
July 2003 **Computer Supported Cooperative Work**, Volume 12 Issue 3
Full text available: Publisher Site Additional Information: full citation, abstract, references, index terms

In this paper, I revisit the concept of recomposition – all the work that development organizations do to make sure that their product fits together and into a broader environment of other technologies. Technologies, such as Configuration Management (CM) systems, can ameliorate some of a software development team's need to engage in recomposition. However, technological solutions do not scale to address other kinds of recomposition needs. This paper focuses on various organizational re ...

Keywords: empirical studies, recomposition, software development

4 Fifteen years of psychology in software engineering: Individual differences and cognitive science
Bill Curtis
March 1984 **Proceedings of the 7th international conference on Software engineering**
Full text available: pdf(943.22 KB) Additional Information: full citation, abstract, references, citings, index terms

Since the 1950's, psychologists have studied the behavioral aspects of software engineering. However, the results of their research have never been organized into a subfield of either software engineering or psychology. This failure results from the difficulty of integrating theory and data from the mixture of paradigms borrowed from psychology. This paper will review some of the psychological research on software engineering performed since the Garmisch Conference in 1968. This review will ...

5 Computing curricula 2001
September 2001 **Journal on Educational Resources in Computing (JERIC)**
Full text available: pdf(613.63 KB) html(2.78 KB) Additional Information: full citation, references, citings, index terms

6 Compiler transformations for high-performance computing
David F. Bacon, Susan L. Graham, Oliver J. Sharp
December 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 4
Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citings, index terms, review

In the last three decades a large number of compiler transformations for optimizing programs have been implemented. Most optimizations for uniprocessors reduce the number of instructions executed by the program using transformations based on the analysis of scalar quantities and data-flow techniques. In contrast, optimizations for high-performance superscalar, vector, and parallel processors maximize parallelism and memory locality with transformations that rely on tracking the properties o ...

Keywords: compilation, dependence analysis, locality, multiprocessors, optimization, parallelism, superscalar processors, vectorization

7 Assessing modular structure of legacy code based on mathematical concept analysis
Christian Lindig, Gregor Snelting
May 1997 **Proceedings of the 19th international conference on Software engineering**
Full text available: pdf(1.79 MB) Additional Information: full citation, references, citings, index terms

Keywords: concept analysis, modularization, reengineering

8 Resourceful systems for fault tolerance, reliability, and safety
Russell J. Abbott
March 1990 **ACM Computing Surveys (CSUR)**, Volume 22 Issue 1
Full text available: pdf(3.36 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Above all, it is vital to recognize that completely guaranteed behavior is impossible and that there are inherent risks in relying on computer systems in critical environments. The unforeseen consequences are often the most disastrous [Neumann 1986]. Section 1 of this survey reviews the current state of the art of system reliability, safety, and fault tolerance. The emphasis is on the contribution of software to these areas. Section 2 reviews current approaches to software fault ...

9 Performance analysis of several back-end database architectures
Robert Brian Hagmann, Domenico Ferrari
March 1986 **ACM Transactions on Database Systems (TODS)**, Volume 11 Issue 1
Full text available: pdf(1.54 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The growing acceptance of database systems makes their performance increasingly more important. One way to gain performance is to off-load some of the functions of the database system to aback-end computer. The problem is what functions should be off-loaded to maximize the benefits of distributed processing. Our approach to this problem consisted of constructing several variants of an existing relational database system. INGRES, that partition the database system software into tw ...

10 Fast and flexible application-level networking on exokernel systems
Gregory R. Ganger, Dawson R. Engler, M. Frans Kaashoek, Héctor M. Briceño, Russell Hunt, Thomas Pinckney
February 2002 **ACM Transactions on Computer Systems (TOCS)**, Volume 20 Issue 1
Full text available: pdf(500.67 KB) Additional Information: full citation, abstract, references, citings, index terms

Application-level networking is a promising software organization for improving performance and functionality for important network services. The Xok/ExOS exokernel system includes application-level support for standard network services, while at the same time allowing application writers to specialize networking services. This paper describes how Xok/ExOS's kernel mechanisms and library operating system organization achieve this flexibility, and retrospectively shares our experiences an ...

Keywords: Extensible systems, OS structure, fast servers, network services

11 Large-scale AOSD for middleware
Adrian Colyer, Andrew Clement
March 2004 **Proceedings of the 3rd international conference on Aspect-oriented software development**
Full text available: pdf(1.82 MB) Additional Information: full citation, abstract, references, citings, index terms

For a variety of reasons, today's middleware systems are highly complex. This complexity surfaces internally in the middleware construction, and externally in the programming models supported and features offered. We believed that aspect-orientation could help with these problems, and undertook a case study based on members of an IBM® middleware product-line. We also wanted to know whether aspect-oriented techniques could scale to commercial project sizes with tens of thousands of classes, m ...

Keywords: aspect-oriented, middleware, refactoring

12 Program decomposition for pointer aliasing: a step toward practical analyses
Sean Zhang, Barbara G. Ryder, William Landi
October 1996 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering**, Volume 21 Issue 6
Full text available: pdf(1.12 MB) Additional Information: full citation, abstract, references, citings, index terms

Pointer aliasing analysis is crucial to compile-time analyses for languages with general-purpose pointer usage (such as C), but many aliasing methods have proven quite costly. We present a technique that partitions the statements of a program to allow separate, and therefore possibly different, pointer aliasing analysis methods to be used on independent parts of the program. This decomposition enables exploration of tradeoff between algorithm efficiency and precision. We also present a new, effi ...

13 Application performance and flexibility on exokernel systems
M. Frans Kaashoek, Dawson R. Engler, Gregory R. Ganger, Héctor M. Briceño, Russell Hunt, David Mazières, Thomas Pinckney, Robert Grimm, John Jannotti, Kenneth Mackenzie
October 1997 **ACM SIGOPS Operating Systems Review , Proceedings of the sixteenth ACM symposium on Operating systems principles**, Volume 31 Issue 5
Full text available: pdf(2.39 MB) Additional Information: full citation, references, citings, index terms

14 Simulating reactive systems by deduction
Yishai A. Feldman, Haim Schneider
April 1993 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 2 Issue 2
Full text available: pdf(3.44 MB) Additional Information: full citation, abstract, references, citings, index terms

Debugging is one of the main uses of simulation. Localizing bugs or finding the reasons for unclear behavior involves going backwards in time, whereas simulation goes forward in time. Therefore, identifying causes with the aid of most existing simulation tools usually requires repeating the simulation several times, each time with reduced holes in the sieve. An alternative is simulation by deduction, a technique in which the steps in the dynamic behavior of the simulated model are deduced b ...

15 Statemate: a working environment for the development of complex reactive systems
D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, a. Shtul-Trauring
April 1988 **Proceedings of the 10th international conference on Software engineering**
Full text available: pdf(1.19 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper provides a brief overview of the STATEMATE system, constructed over the past three years by i-Logix Inc., and Ad Cad Ltd. STATEMATE is a graphical working environment, intended for the specification, analysis, design and documentation of large and complex reactive systems, such as real-time embedded systems, control and communication systems, and interactive software. It enables a user to prepare, analyze and debug diagrammatic, yet precise, descriptions of the system under devel ...

16 Formalizing space shuttle software requirements: four case studies
Judith Crow, Ben Di Vito
July 1998 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 7 Issue 3
Full text available: pdf(267.77 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This article describes four case studies in which requirements for new flight software subsystems on NASA's Space Shuttle were analyzed using mechanically supported formal methods. Three of the studies used standard formal specification and verification techniques, and the fourth used state exploration. These applications illustrate two thesis: (1) formal methods complement conventional requirements analysis processes effectively and (2) formal methods confer benefits even when only selecti ...

Keywords: flight software, formal methods, requirements analysis, space shuttle, state exploration, theorem proving

17 Intelligent Search Methods for Software Maintenance
Huixiang Liu, Timothy C. Lethbridge

December 2002 **Information Systems Frontiers**, Volume 4 Issue 4

Full text available: Publisher Site Additional Information: full citation, abstract, index terms

This paper describes a study of what we call *intelligent search techniques* as implemented in a software maintenance environment. The techniques studied include abbreviation concatenation and abbreviation expansion. We also describe rating algorithms used to prioritize the query results. To evaluate our approach, we present a series of experiments in which we compare our algorithms' ratings of results to ratings provided by software engineers.

Keywords: abbreviation expansion, information retrieval, intelligent search, software engineering tools, software maintenance

18 "Topologies"—distributed objects on multicomputers
Karsten Schwan, Win Bo
May 1990 **ACM Transactions on Computer Systems (TOCS)**, Volume 8 Issue 2

Full text available: pdf(3.83 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Application programs written for large-scale multicomputers with interconnection structures known to the programmer (e.g., hypercubes or meshes) use complex communication structures for connecting the applications' parallel tasks. Such structures implement a wide variety of functions, including the exchange of data or control information relevant to the task computations and/or the communications required for task synchronization, message forwarding/filtering under program control, and so o ...

19 A structural view of the Cedar programming environment
Daniel C. Swinehart, Polle T. Zellweger, Richard J. Beach, Robert B. Hagmann
August 1986 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 8 Issue 4

Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents an overview of the Cedar programming environment, focusing on its overall structure—that is, the major components of Cedar and the way they are organized. Cedar supports the development of programs written in a single programming language, also called Cedar. Its primary purpose is to increase the productivity of programmers whose activities include experimental programming and the development of prototype software systems for a high-performance personal computer. T ...

20 An object-oriented approach to automated generation of challenge examinations using Ada 95
Arthur Irving Littlefield
January 1997 **ACM SIGAda Ada Letters**, Volume XVII Issue 1

Full text available: pdf(1.04 MB) Additional Information: full citation, abstract, index terms

The primary objective of this paper is to analyze and evaluate the usefulness of object-oriented development and the Ada 95 programming language as applied to a specific software development project. A secondary objective is to show that structured development is still useful while applying object-oriented development and that the two methods can be integrated. The project is to develop an automated tool for generation of challenge examinations to test the knowledge of students in a given subjec ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

Yahoo! My Yahoo! Mail Welcome, **Guest** [Sign In] Search Home Help

Web | Images | Directory | Yellow Pages | News | Products

are test modular decomposition Search

Shortcuts Advanced Search Preferences

Search Results Results **21 - 40** of about **22,700** for **software test modular decomposition** . Search took 0.11 seconds. (Abo this page...

21. Totally Data-Driven Automated Testing (PDF)
... **test** tools, not. testing **software**.Slide 8The "Functional**Decomposition**" MethodReduce **test** cases to their ... a st
www.dkl.com/pdf/datadriven.pdf - 470k - View as html - More pages from this site

22. Automated Testing Methodologies
... the "Functional **Decomposition**" script development methodology is to reduce all **test** cases to ... This allows an a
www.sqa-test.com/method.html - 79k - Cached - More pages from this site

23. DETAILED SYLLABUS 5 & 6
... **software** process - interface specification - behavior specification - architectural design - system structuring - cont
www.cednss.org/SYLLABUS%2056.htm - 405k - Cached - More pages from this site

24. Chapter 1 **Software** Engineering Principles (PDF)
1 **Software** Engineering PrinciplesChapter 1**Software** Engineering PrinciplesTopics**Software** Life CycleModules anc ExampleProgram Testing1.1. ... Stepwise Refinement Problem **decomposition** is based on ... The class and **test** dr
www.ugrad.cs.ubc.ca/~cs252/notes.pdf - 1602k - View as html - More pages from this site

25. http://academic.cuesta.edu/jdalbey/outcomes.htm
... the fundamental hardware and **software** components of a computer system ... Create **test** cases for iterative cont
academic.cuesta.edu/jdalbey/outcomes.htm - 8k - Cached - More pages from this site

26. RSP&A **software** engineering glossary
... **modular** building block for computer **software**. Component reuse - the ability to reuse a portion of a model, sourc
www.rspa.com/spi/glossary.html - 28k - Cached - More pages from this site

27. General Principles of **Software** Validation; Final Guidance for Industry and FDA Staff
... requirements specification, **software** design specification, **software test** specification, **software** ... **modular** struc
www.fda.gov/cdrh/comp/guidance/938.html - 135k - Cached - More pages from this site

28. Doug Moen
... Programming Paradigms and Methodologies. **Modular decomposition**, object oriented programming, multi-threa
www.moens.org/doug/resume.html - 10k - Cached - More pages from this site

29. Skynet Innovations Pvt. Ltd. - Application Development Methodology
... purpose of the **Modular Decomposition** Phase is to produce a detailed design of the **software** components base
www.skyinfonet.com/html/devp_custm.shtml - 33k - Cached - More pages from this site

30. Glossary M
... Metric based **test** data generation. (NBS) The process of generating **test** sets for ... **Modular decomposition**. A
www.validationstation.com/glossary/glossarym.htm - 36k - Cached - More pages from this site

31. 4.0 GEOS-3 **Software** Development
... 4.3.1.1 **Test** Planning During **Software** Requirements and Design ... Fortran 90 provides a **modular** programming
ess.gsfc.nasa.gov/lys/SDP4 - 56k - Cached - More pages from this site

32. NISTIR 4909: **Software** Quality Assurance: Documentation and Reviews
... experience in **software** engineering and **software** quality, working ... in **software test** types, and the relationship
hissa.nist.gov/publications/nistir4909 - 126k - Cached - More pages from this site

33. EE Times -Strategies for Implementing Embedded Network Processing **Software**
... Figure 1 depicts the **decomposition** of the Control Plane into two major ... be utilized to **test** the **modular** blocks
www.eetimes.com/story/OEG20021106S0015 - 69k - Cached - More pages from this site

34. http://www.cis.drexel.edu/faculty/gasson/courses/isys200s/Wk9.ppt (MICROSOFT POWERPOINT)
... eXtreme Programming and other agile **software** development approaches. What is ... **Modular Decomposition**. F
www.cis.drexel.edu/faculty/gasson/courses/isys200s/Wk9.ppt - 423k - View as html - More pages from this site

35. Intra-**Modular** Structuring in Model-Oriented Specification: Expressing Non-interference with Rea
Compositionality provides the key to managing complexity in **software** systems and thus should be sought at all leve
decomposition of system specifications into ...
citeseer.ist.psu.edu/518873.html - 25k - Cached - More pages from this site

36. QWE'99 -- Speaker Biographies
QWE'99) 1-5 November 1999, Brussels, Belgium. SPEAKER BIOGRAPHIES ... of Worldwide **Software Test** and An
decomposition methods for the RT-Tester **test** specification ...
www.soft.com/QualWeek/QWE99/qwe99.bios.html - 133k - Cached - More pages from this site

37. **Software** Testing Technology
Software Testing Technology. Overview. INTRODUCTION. This section provides a high-level overview and perspec
many cases, triple **modular** redundancy is provided at the ... as an "acceptance **test**" process. A **software** system ca
www.ee.cooper.edu/courses/course_pages/past_courses/EE352/TESTING.html - 91k - Cached - More pages from t

38. Gorilla.it: Product: 'THE ART AND SCIENCE OF C'
... Econ. Computer Dict./ref./**test** Prep Earth Sci ... uses standard **software** engineering strategies - procedural abstr
www.gorilla.it/gorilla/product.asp?sku=0201543222&dept_id= - 17k - Cached - More pages from this site

39. GLOSSARY OF COMPUTERIZED SYSTEM AND **SOFTWARE** DEVELOPMENT TERMINOLOG
GLOSSARY OF COMPUTERIZED SYSTEM AND **SOFTWARE** DEVELOPMENT TERMINOLOGY. Note: This docu
development and computerized systems ... The design **decomposition** of the **software** item; e ... design; **software**
www.fda.gov/ora/inspect_ref/igs/gloss.html - 212k - Cached - More pages from this site

40. **software** management by colyer
... depot, which allows **modular** management of **software** collections ... A straightforward **decomposition** of /usr/loc
andrew2.andrew.cmu.edu/depot/AFSUG-2.html - 33k - Cached - More pages from this site

Results Page:
Prev 1 2 3 4 5 6 7 8 9 10 **Next**

Web | Images | Directory | Yellow Pages | News | Products
Your Search: software test modular decomposition Search

Help us improve your search experience. Send us feedback.
One-click to Mail, Search and More! - Yahoo! Toolbar

Web Images Groups News Froogle **more »**

incremental fault detection software debug Search Advanced Search Preferences

Web Results **11** - **20** of about **11,100** for **incremental fault detection software debug**. (**0.75** seconds)

[PDF] Hardware Support for Reliability
File Format: PDF/Adobe Acrobat - View as HTML
... If an error or **fault** occurs: – Invalidate ... communication Final **incremental** undo, possibly fix ... productivity – Enhances **Software Debugging** (**detection**, analysis ...
www-courses.cs.uiuc.edu/~cs433/notes/reliable1.pdf - Similar pages

[PPT] Defect testing
File Format: Microsoft Powerpoint 97 - View as HTML
... and analysis of the program for error **detection**. ... may be an indication of **faults** in the ... Cleanroom development process depends on **incremental** development, static ...
se.math.spbu.ru/Courses/Testing/SoftwareInspections.ppt - Similar pages

ACM Transactions on **Software** Engineering and Methodology (to ...
... Brown, Gould - 1987 28 An **incremental** version of ... of test set minimization on **fault detection** effective ... 1998 25 IEEE Transactions on **Software** Engineering (context ...
citeseer.ist.psu.edu/671671.html - 25k - Cached - Similar pages

Citations: **Software** Implemented **Fault** Tolerance: Technologies and ...
... since they support selective and **incremental** state saving ... reusable components for automatic **detection** and recovery ... which varying levels of **fault** tolerance are ...
citeseer.ist.psu.edu/context/17944/0 - 36k - Cached - Similar pages
[More results from citeseer.ist.psu.edu]

Optimal tracing and **incremental** reexecution for **debugging** long ...
... Optimal tracing and **incremental** reexecution for **debugging** long ... of **Fault** Tolerant Computing Systems, (1992). ... Crummey , TJ LeBlanc, A **software** instruction counter ...
portal.acm.org/citation.cfm?id=773473.178477 - Similar pages

[PDF] NT-SwiFT: **Software** Implemented **Fault** Tolerance on Windows NT
File Format: PDF/Adobe Acrobat - View as HTML
... makes the implementa- tion of some **fault** tolerance mechanisms ... to deal with process thread failure **detection** and recov- ery, **incremental** state checkpoint ...
www.usenix.org/publications/library/ proceedings/usenix-nt98/full_papers/huang/huang.pdf - Similar pages

[PDF] Using Neural Networks for **Fault** Diagnosis
File Format: PDF/Adobe Acrobat - View as HTML
... is to set a **fault** in the **software**, and let ... As to BP based FD, 20 instances **incremental** learning means ... Lee J, Tsai J. On-line **fault detection** using integrated ...
cs.nju.edu.cn/people/zhouzh/ zhouzh.files/publication/ijcnn00b.pdf - Similar pages

[PDF] **Software Fault** Tolerance of Distributed Programs Using Computation ...
File Format: PDF/Adobe Acrobat - View as HTML
... Keywords: predicate **detection**, testing and **debugging**, **software**-**fault** tolerance, pruning search-space, partial- order methods 1. Introduction Writing ...
www.utdallas.edu/~neerajm/publications/13/icdcs03.pdf - Similar pages

EEProductCenter.com :: Press Release :: Aptix Announces Shipment ...
... eg 2x) – Explorer(TM) **software** enhancements – Improved ... FPGA modules (setup and **incremental** changes ... Increased hardware coverage, reduced time to **detect faults**. ...
www.eeproductcenter.com/ showPressRelease.jhtml?articleID=87533 - 26k - Cached - Similar pages

Jiang Brandon Liu's Résumé
... Amiri and Andreas Veneris "**Incremental Fault** Diagnosis and ... Network and Host security;

intrusion **detection**. ... and implementation of the following **software** in C ...
www.eecg.toronto.edu/~liuji/ - 19k - Cached - Similar pages

◀ Gooooooooooogle ▶
Result Page: **Previous** 1 2 3 4 5 6 7 8 9 10 11 **Next**

incremental fault detection software | Search

Search within results | Language Tools | Search Tips

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ○ The ACM Digital Library ● The Guide

software debug by decomposition

THE GUIDE TO COMPUTING LITERATURE

Feedback Report a problem Satisfaction survey

Terms used **software debug by decomposition**

Found **102,083** of **825,846**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search
Try this search in The Digital Library

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale ☐ ☐ ☐ ▨ ▨

1 Computing curricula 2001
September 2001 **Journal on Educational Resources in Computing (JERIC)**
Full text available: pdf(613.63 KB) html(2.78 KB)
Additional Information: full citation, references, citings, index terms

2 Simulating reactive systems by deduction
Yishai A. Feldman, Haim Schneider
April 1993 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 2 Issue 2
Full text available: pdf(3.44 MB)
Additional Information: full citation, abstract, references, citings, index terms

Debugging is one of the main uses of simulation. Localizing bugs or finding the reasons for unclear behavior involves going backwards in time, whereas simulation goes forward in time. Therefore, identifying causes with the aid of most existing simulation tools usually requires repeating the simulation several times, each time with reduced holes in the sieve. An alternative is simulation by deduction, a technique in which the steps in the dynamic behavior of the simulated model are deduced b ...

3 Jargons for domain engineering
Lloyd H. Nakatani, Mark A. Ardis, Robert G. Olsen, Paul M. Pontrelli
December 1999 **ACM SIGPLAN Notices , Proceedings of the 2nd conference on Domain-specific languages**, Volume 35 Issue 1
Full text available: pdf(886.76 KB)
Additional Information: full citation, abstract, references, citings, index terms

In the Family-oriented Abstraction, Specification and Translation (FAST) domain engineering process for software production, a member of a software product family is automatically generated from a model expressed in a DSL. In practice, the time and skill needed to make the DSLs proved to be bottlenecks. FAST now relies on jargons, a kind of easy-to-make DSL that domain engineers who are not language experts can quickly make themselves. We report our experiences with jargons in the FAST proc ...

4 Using laboratories to teach software engineering principles in the introductory computer science curriculum
James Robergé, Candice Suriano
March 1994 **ACM SIGCSE Bulletin , Proceedings of the twenty-fifth SIGCSE symposium on Computer science education**, Volume 26 Issue 1
Full text available: pdf(423.05 KB)
Additional Information: full citation, abstract, references, citings, index terms

If students are to internalize software engineering concepts and incorporate them into their individual software development styles, they must use these concepts during the initial stages of their computer science education. In this paper, we examine how laboratories that emphasize software development can be used to familiarize students with the basic elements of software engineering during the introductory computer science course sequence.

5 Statemate: a working environment for the development of complex reactive systems
D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, a. Shtul-Trauring
April 1988 **Proceedings of the 10th international conference on Software engineering**

Full text available: pdf(1.19 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper provides a brief overview of the STATEMATE system, constructed over the past three years by i-Logix Inc., and Ad Cad Ltd. STATEMATE is a graphical working environment, intended for the specification, analysis, design and documentation of large and complex reactive systems, such as real-time embedded systems, control and communication systems, and interactive software. It enables a user to prepare, analyze and debug diagrammatic, yet precise, descriptions of the system under devel ...

6 Program decomposition for pointer aliasing: a step toward practical analyses
Sean Zhang, Barbara G. Ryder, William Landi
October 1996 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering**, Volume 21 Issue 6

Full text available: pdf(1.12 MB) Additional Information: full citation, abstract, references, citings, index terms

Pointer aliasing analysis is crucial to compile-time analyses for languages with general-purpose pointer usage (such as C), but many aliasing methods have proven quite costly. We present a technique that partitions the statements of a program to allow separate, and therefore possibly different, pointer aliasing analysis methods to be used on independent parts of the program. This decomposition enables exploration of tradeoff between algorithm efficiency and precision. We also present a new, effi ...

7 Visualising and debugging distributed multi-agent systems
Divine T. Ndumu, Hyacinth S. Nwana, Lyndon C. Lee, Jaron C. Collis
April 1999 **Proceedings of the third annual conference on Autonomous Agents**

Full text available: pdf(1.14 MB) Additional Information: full citation, references, citings, index terms

8 Recomposition: Coordinating a Web of Software Dependencies
Rebecca E. Grinter
July 2003 **Computer Supported Cooperative Work**, Volume 12 Issue 3

Full text available: Publisher Site Additional Information: full citation, abstract, references, index terms

In this paper, I revisit the concept of recomposition – all the work that development organizations do to make sure that their product fits together and into a broader environment of other technologies. Technologies, such as Configuration Management (CM) systems, can ameliorate some of a software development team's need to engage in recomposition. However, technological solutions do not scale to address other kinds of recomposition needs. This paper focuses on various organizational re ...

Keywords: empirical studies, recomposition, software development

9 Compiler transformations for high-performance computing
David F. Bacon, Susan L. Graham, Oliver J. Sharp
December 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 4

Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citings, index terms, review

In the last three decades a large number of compiler transformations for optimizing programs have been implemented. Most optimizations for uniprocessors reduce the number of instructions executed by the program using transformations based on the analysis of scalar quantities and data-flow techniques. In contrast, optimizations for high-performance superscalar, vector, and parallel processors maximize parallelism and memory locality with transformations that rely on tracking the properties o ...

Keywords: compilation, dependence analysis, locality, multiprocessors, optimization, parallelism, superscalar processors, vectorization

10 Solving systems of partial differential equations using object-oriented programming techniques with coupled heat and fluid flow as example
Hans Petter Langtangen, Otto Munthe
March 2001 **ACM Transactions on Mathematical Software (TOMS)**, Volume 27 Issue 1
Full text available: pdf(1.01 MB) Additional Information: full citation, abstract, references, index terms

This paper exploits object-oriented implementation techniques to facilitate the development computer codes for solving systems of coupled partial differential equations. We show how to build a simulator for equation systems by merging independent solvers for each equation that enters the system. The main goal is to obtain a rapid, robust, and reliable software development process with extensive reuse of implemented code. Coupled heat and fluid flow in pipes is used as example for illustrati ...

Keywords: C++, coupled heat-fluid, diffpack, finite elements, non-Newtonian fluids, object-oriented programming, software development, systems of partial differential equations

11 A window based visual debugger for a real time Ada tasking environment
Jeff Gilles, Ray Ford
July 1988 **Proceedings of the fifth Washington Ada symposium on Ada**
Full text available: pdf(922.10 KB) Additional Information: full citation, references, citings, index terms

12 When hardware becomes software: designing a safety-critical system with Ada
James Hummer, Loïc Briand
December 1992 **Proceedings of the conference on TRI-Ada '92**
Full text available: pdf(748.30 KB) Additional Information: full citation, references, index terms

13 Scalable software libraries
Don Batory, Vivek Singhal, Marty Sirkin, Jeff Thomas
December 1993 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 1st ACM SIGSOFT symposium on Foundations of software engineering**, Volume 18 Issue 5
Full text available: pdf(861.51 KB) Additional Information: full citation, abstract, references, citings, index terms

Many software libraries (e.g., the Booch C++ Components, libg++, NIHCL, COOL) provide components (classes) that implement data structures. Each component is written by hand and represents a unique combination of features (e.g. concurrency, data structure, memory allocation algorithms) that distinguishes it from other components.We argue that this way of building data structure component libraries is inherently unscalable. Libraries should not enumerate complex components with numerous features; ...

14 The use of program dependence graphs in software engineering
Susan Horwitz, Thomas Reps
June 1992 **Proceedings of the 14th international conference on Software engineering**
Full text available: pdf(2.58 MB) Additional Information: full citation, references, citings, index terms

15 Fast hardware-software coverification by optimistic execution of real processor
Sungjoo Yoo, Jong-Eun Lee, Jinyong Jung, Kyungseok Rha, Youngchul Cho, Kiyoung Choi
January 2000 **Proceedings of the conference on Design, automation and test in Europe**
Full text available: pdf(102.05 KB) Publisher Site Additional Information: full citation, references, index terms

16 Extracting concepts from file names: a new file clustering criterion
Nicolas Anquetil, Timothy Lethbridge
April 1998 **Proceedings of the 20th international conference on Software engineering**
Full text available: pdf(1.11 MB) Publisher Site Additional Information: full citation, references, citings, index terms

17 Workshop on compositional software architectures: workshop report
May 1998 **ACM SIGSOFT Software Engineering Notes**, Volume 23 Issue 3
Full text available: pdf(2.91 MB) Additional Information: full citation, index terms

18 Fifteen years of psychology in software engineering: Individual differences and cognitive science
Bill Curtis
March 1984 **Proceedings of the 7th international conference on Software engineering**
Full text available: pdf(943.22 KB) Additional Information: full citation, abstract, references, citings, index terms

Since the 1950's, psychologists have studied the behavioral aspects of software engineering. However, the results of their research have never been organized into a subfield of either software engineering or psychology. This failure results from the difficulty of integrating theory and data from the mixture of paradigms borrowed from psychology. This paper will review some of the psychological research on software engineering performed since the Garmisch Conference in 1968. This review will ...

19 Software engineering #1: Assessing the complexity of software architecture
Mohsen AlSharif, Walter P. Bond, Turky Al-Otaiby
April 2004 **Proceedings of the 42nd annual Southeast regional conference**
Full text available: pdf(334.88 KB) Additional Information: full citation, abstract, references, index terms

A central activity of software architecture design is decomposing the system into subsystems (i.e. components) that work together to satisfy the required functionality. The purpose of this activity is to reduce problem complexity into smaller manageable parts. Complexity can never be totally eliminated; however the designer/architect can reduce it.The decomposition process is an art form; the architect must decide whether to assign a specific functionality to a given component or to defer some o ...

Keywords: Full Function Points, architecture assessment, architecture complexity, coupling, software architecture

20 Software engineering techniques in design automation&madash;a tutorial
Robert J. Smith
January 1977 **Proceedings of the 14th conference on Design automation**
Full text available: pdf(919.98 KB) Additional Information: full citation, abstract, references, index terms

Several useful software engineering techniques, disciplines and perspectives are related to typical software development problems in design automation. Specific examples from recent experience illustrate both beneficial and undesirable practices. Oriented toward managers and practicing software engineers, the tutorial discusses system structure, control and data structures, programming guidelines, work habits, testing, documentation and operational maintenance.

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next